Abbiamo davvero bisogno del pensiero computazionale?

Michael Lodi, Simone Martini, Enrico Nardelli

Abbiamo davvero bisogno del pensiero computazionale?[1]

Michael Lodi, Simone Martini, Enrico Nardelli

Sommario*: Esaminiamo in che misura sia necessaria l'espressione "pensiero computazionale" ed argomentiamo che può essere controproducente usarla in modo eccessivo insistendo a cercarne una definizione operativamente precisa. Questo termine dovrebbe servire, in modo strumentale, per spiegare sinteticamente perché l'informatica è una disciplina scientifica nuova e distinta dalle altre e perché dovrebbe essere insegnata a scuola a tutti i livelli.*

Abstract*: We discuss if and to what extent we need the expression "computational thinking". We argue that looking for a precise, operational definition could be misleading, and that this expression should not be used everywhere instead of "informatics". "Computational thinking", instead, is an instrumental expression which may help in suggesting that informatics is a new and distinct scientific discipline, and in motivating why it should be taught in the schools of any level.*

Keywords: informatics, computational thinking, school, mathematics, teaching

Vogliamo analizzare in che misura sia necessaria l'espressione "pensiero computazionale" (CT, nel seguito, dall'espressione inglese "*computational thinking*"), resa popolare da un importante articolo di Jeannette Wing [1], seguito da una imponente discussione con centinaia di articoli successivi. Non esiste ancora una definizione operativa universalmente accettata di tale espressione: discuteremo nel seguito se ne abbiamo davvero bisogno e per quale scopo.

Chiariamo subito che, da informatici, siamo profondamente convinti che sia necessario il **concetto** di pensiero computazionale, inteso come "essere in grado di pensare come un informatico ed essere in grado di applicare questa competenza ad ogni settore dell'attività umana". Anticipando le conclusioni, tuttavia, probabilmente il termine è più necessario come strumento, come un modo abbreviato di riferirsi ad un concetto ben strutturato, mentre potrebbe essere controproducente insistere a cercarne una definizione operativamente precisa. L'espressione è utile per indicare che l'informatica è una disciplina scientifica nuova e distinta dalle altre. E per questo scopo cercheremo di dare una definizione un po' più generale di quella della Wing.

1. Perché parliamo di pensiero computazionale

Iniziamo dal principio. Wing introduce in [1] il termine CT per sostenere l'importanza di insegnare ad ogni studente «*come pensa un informatico*» . In questo lavoro non fornisce una definizione, che viene poi esplicitata in lavori successivi, in collaborazione con Cuny e Snyder[2]: «*CT è l'insieme dei processi mentali usati per formulare i problemi e le loro soluzioni in modo tale che la descrizione*

[2] Questa formulazione, da [2], si riferisce ad un "articolo in preparazione" mai pubblicato: "Demystifying Computational Thinking for Non-Computer Scientists" (2010). Inoltre, la Wing evidenzia come essa sia originata da una discussione con Alfred Aho, che fornisce una definizione molto simile (sebbene più incentrata sul "pensiero algoritmico") in [3]

delle soluzioni sia effettivamente eseguibile da un agente che elabora informazioni». Nel far questo, segue la strada che altri avevano tracciato prima di lei. Donald Knuth, ben conosciuto sia dai matematici che dagli informatici, nel 1974 scrive [4] «*A dire il vero, una persona non comprende* davvero *un argomento se non quando riesce ad insegnarlo ad un* computer».

George Forsythe, già presidente dell'ACM[3] ed uno dei padri fondatori dei corsi di studio universitari in informatica, nel 1968 aveva scritto [5]: «*L'apprendimento più valido nell'istruzione tecnica o scientifica è rappresentato da quegli strumenti mentali di utilità generale che rimangono validi per tutta la vita. Considero il linguaggio naturale e la matematica come i più importanti fra questi strumenti, e l'informatica come il terzo*».

Tuttavia, la popolarità acquisita dall'espressione "pensiero computazionale" con l'articolo di Wing rischia di rovinare il suo scopo originario. Sempre più persone considerano il CT come un nuovo soggetto di insegnamento, concettualmente diverso o distinto dall'informatica. Nella ricerca di individuare quella definizione operativa che la Wing non ha fornito nel suo articolo, viene evidenziato questo o quell'altro aspetto (l'astrazione, la ricorsività, la risoluzione dei problemi, …), e così facendo se ne oscura il significato più importante: uno scenario discusso in modo chiaro e rivelatore da Armoni [6] e Denning [7].

La situazione si ingarbuglia ancor di più nel settore dell'istruzione scolastica. Parlare di CT nell'ambito della scuola è un atteggiamento rischioso: i filosofi, giustamente, si chiedono cosa intendiamo dire parlando di "insegnare il pensiero"; i matematici osservano che molti aspetti del CT (algoritmo, astrazione, ricorsività, problem solving, …) sono anche elementi tipici della matematica (che, però, non viene chiamata per questo "pensiero matematico"); i pedagogisti chiedono su quali basi siamo sicuri che sia davvero efficace l'insegnamento del CT; gli insegnanti vogliono sapere quali sono i metodi e gli strumenti per insegnare questa nuova disciplina e come possono imparare ad insegnarla; infine, i genitori sono da un lato contenti perché finalmente la scuola si allinea ad una società sempre più digitale, ma dall'altro sono preoccupati di ciò che accadrà ai loro figli nel futuro se imparano a programmare coi linguaggi di oggi, che saranno obsoleti tra qualche anno. Per non parlare, poi, di quelli che vorrebbero usare l'espressione CT senza alcun riferimento all'articolo originale della Wing o a concetti fondamentali dell'informatica, e che si sentono così liberi di attribuirgli, di volta in volta, i significati più diversi.

Riteniamo che l'articolo originale della Wing avesse solo lo scopo di "indicare la direzione": occorre insegnare i concetti fondamentali dell'informatica nella scuola, possibilmente fin dalle prime classi, perché quei concetti sono oggi importanti quanto quelli della matematica, della fisica, della letteratura, della storia delle idee. Siamo anche convinti che in prima approssimazione il significato dell'espressione «*come pensa un informatico*» possa essere sufficientemente chiarito dalla spiegazione auto-referenziale "*Il CT è l'insieme delle competenze mentali e cognitive ottenute mediante lo studio e la pratica dell'informatica*": la "conoscenza tacita" definita da Polanyi [8]. Forniremo tuttavia nella sezione 3 alcuni utili approfondimenti su tale espressione, ricordando comunque che già Knuth nel 1974 aveva avvisato, nel discutere il termine "informatica", che «*i concetti sottostanti sono più importanti del nome*» [4]. Questo è ancor più vero, crediamo, per il

[3] La *Association for Computing Machinery* (ACM) è la più grande associazione di informatici, sia scienziati che professionisti, esistente al mondo.

CT. Ciò che importa davvero è che l'informatica in quanto disciplina scientifica sia insegnata da subito nella scuola. E in effetti questa è la strada seguita da alcuni grandi Paesi, di cui presenteremo i tre casi più rilevanti.

In Inghilterra, il programma nazionale di studio relativo all'informatica[4], pubblicato dal Ministero dell'Istruzione nel Settembre 2013 ed obbligatorio per tutte le scuole a partire dall'a.s. 2014-15, utilizza il CT nel senso sopra presentato di "ciò che si ottiene dallo studio e dalla pratica dell'informatica". Infatti, usa il termine "pensiero computazionale" solo nella frase iniziale «*Un'istruzione di elevata qualità nell'informatica fornisce agli allievi la capacità di usare il pensiero computazionale e la creatività per comprendere e cambiare il mondo*» e poi in un paio di altre occasioni, negli obiettivi per il Livello 3 ("*Key Stage 3*") «*comprendere diversi algoritmi fondamentali che rispecchiano il pensiero computazionale*» e per il livello 4 «*sviluppare e applicare le abilità analitiche, di risoluzione dei problemi, di progettazione e di pensiero computazionale*». Il curriculum non definisce mai il termine. Osserviamo subito, nella prima citazione, il legame tra CT e creatività, un tema che riprenderemo più avanti in modo esteso.

Negli Stati Uniti, l'atto legislativo "*Every Student Succeeds Act*" (ESSA), approvato dal Congresso nel 2015 con supporto bipartisan, ha introdotto l'informatica tra le "materie per un'istruzione a tutto tondo" (*well rounded educational subject*), che devono essere insegnate nella scuola «*con lo scopo di fornire a tutti gli studenti l'accesso ad un curriculum ed un'esperienza formativa migliori*» e non fa cenno al CT. Nel gennaio 2016, l'allora Presidente Obama ha lanciato l'iniziativa "*Computer Science For All*"[5] il cui obiettivo è «*di mettere tutti gli studenti americani, dall'asilo al liceo, in condizione di imparare l'informatica ed essere così attrezzati con le abilità di pensiero computazionale di cui hanno bisogno …*». Di nuovo, il CT è ciò che si ottiene una volta che si è appresa l'informatica.

In Francia, la "*Académie des Sciences*", la più importante istituzione rappresentante gli scienziati di quel paese, ha pubblicato nel Maggio 2013 il rapporto "*L'insegnamento dell'informatica in Francia. È urgente non aspettare oltre*"[6], che raccomanda – a proposito dell'insegnamento dell'informatica – «*l'insegnamento dovrebbe iniziare nella primaria, attraverso l'esposizione alle nozioni di base dell'informatica e degli algoritmi, … <e> dovrebbe essere ulteriormente sviluppato nella scuola media e nella superiore*». Analizzando il loro uso di CT ("*pensée informatique*") è chiaro che anche nella loro visione il termine denota la specifica modalità di pensiero sviluppata dall'apprendimento dell'informatica. Solo un paio di esempi: «*l'informatica … conduce ad un differente modo di pensare, chiamato CT*» e «*imparare la programmazione è un modo di scoprire i rudimenti del CT*».

Da questi tre autorevoli esempi risulta chiaramente che il CT non è una nuova materia d'insegnamento: ciò che deve essere insegnato nelle scuole è l'informatica e il CT è, al più, il sedimento concettuale di quell'insegnamento, quello che resta anche quando gli aspetti tecnici sono stati dimenticati.

---

[4] https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

[5] http://www.nsf.gov/csforall

[6] http://www.academie-sciences.fr/pdf/rapport/rads_0513.pdf

D'altro canto, molti lavori che fanno riferimento al CT nel titolo o nel sommario (la sola Digital Library dell'ACM ne riporta più di 400) sembrano sostenere che il CT è qualcosa di nuovo e di diverso. Alcuni addirittura sostengono che il "coding" (che considerano diverso dalla "programmazione") è tutto quello che serve per impararlo! Il recente [9] discute i rischi e i limiti di questo approccio.

Siamo convinti che questa separazione tra CT e informatica sia sbagliata e fuorviante e che sul lungo periodo rischi di creare più danni che benefici. In fin dei conti, nelle scuole non viene insegnato il "pensiero linguistico" o il "pensiero matematico" e non ci sono i "programmi scolastici" o i "metodi di valutazione" per queste materie. Semplicemente, vengono insegnati "Italiano" e "Matematica", e ne vengono valutate le competenze. In seguito, le competenze linguistiche (o matematiche) acquisite con lo studio dell'Italiano (o della Matematica), oltre ad essere usate in quanto tali, sedimentano in un apparato concettuale di categorie fondamentali per l'interpretazione della realtà, oltre a trovare ulteriore utilizzo in altre materie. Tra il CT e l'informatica vi è la stessa relazione. Pertanto dovremmo discutere cosa insegnare e come valutare le competenze di Informatica nella scuola, e non discutere come insegnare e valutare le competenze di CT.

In sintesi, parlare di CT ha lo scopo di aiutare a comprendere che **non stiamo parlando di sistemi e strumenti, ma di princìpi e metodi**: ci stiamo focalizzando sui concetti scientifici fondamentali. Diversamente da quanto accade con l'italiano o la matematica, siamo costretti ad esplicitare questa distinzione, dal momento che l'informatica ha anche un'imponente componente tecnologica che la fa, da molti, identificare con i computer e il loro uso strumentale.

2. Perché l'informatica non è la matematica

È ovviamente lecito, a questo punto, chiedere quali siano questi concetti scientifici fondamentali e se è possibile fornire qualche esempio. Si tratta di un passaggio critico per spiegare perché l'informatica è una nuova disciplina scientifica e qual è il suo valore educativo.

Spiegare in che senso "come pensa un informatico" è differente da "come pensa un matematico" è un punto assai importante. Knuth aveva fornito un esempio brillante (anche se non di immediata comprensione) già nel suo articolo del 1974. Riguarda il problema di trovare il "massimo comun divisore destro" di due matrici di $n \times n$ numeri interi. La risposta di un matematico è: «*Sia* ***R*** *l'anello delle matrici di interi; in questo anello la somma dei due ideali principali sinistri è principale, quindi sia D tale che* ***R*** *A +* ***R*** *B =* ***R*** *D. Allora D è il massimo comun divisore destro di A e B*» [4]. Questa risposta è chiaramente insoddisfacente per l'informatico, per cui **una soluzione è ciò che viene restituito da un *processo* che calcola la risposta e non da un'*equazione* che definisce la risposta**. Abbiamo intenzionalmente usato il termine "processo" al posto del più usuale "algoritmo" per sottolineare il fatto che abbiamo un "processo" soltanto quando l'algoritmo è stato tradotto in un opportuno "linguaggio" e il codice ottenuto è in esecuzione da parte di un "automa". In tal modo, tre dei pilastri fondamentali dell'informatica (algoritmo, linguaggio, automa) sono tutti coinvolti nel caratterizzare la differenza tra il punto di vista del matematico e quello dell'informatico.

Ogni scienza ha il suo particolare punto di vista sul mondo, il suo "paradigma concettuale" in base al quale descrive e spiega i fenomeni. Così - giusto per fare alcuni esempi - sono concetti essenziali per un matematico la quantità e le loro relazioni, per un fisico le forze e le masse, per un biologo l'organismo, il metabolismo e la riproduzione, per un chimico le molecole e le reazioni, e così via. Ognuno di questi paradigmi può essere usato per descrivere la stessa realtà, ed a seconda dei casi e dei contesti uno di essi può essere quello più utile per la comprensione e la spiegazione oppure può esserlo la loro combinazione. Concetti quali quello di algoritmo, linguaggio, automa fanno parte del paradigma concettuale dell'informatico e possono essere impiegati con successo per fornire descrizioni complementari a quelle fornite da altre scienze. Ne è un esempio lo studio dei sistemi viventi (p.es. una cellula), nella quale le tradizionali descrizioni del biologo e del chimico sono state affiancate prima da quelle dei matematici e dei fisici e poi da quelle degli informatici, ottenendo, con l'insieme delle rappresentazioni, un quadro notevolmente più completo.

Un esempio paradigmatico del modo di descrivere il mondo da parte dell'informatica (cioè del CT in azione) è esemplificato da una sequenza[7] tratta dal film "Apollo 13". Gli astronauti si trovano in una situazione critica a bordo del modulo lunare, e devono essere guidati da terra per costruire un filtro per il sistema di purificazione dell'aria. In questo caso, gli esperti del centro di controllo devono ragionare come informatici per far sì che un esecutore sia in grado di comprendere ed eseguire le istruzioni che gli vengono fornite in modo che possa essere raggiunto l'obiettivo desiderato. Qui l'esecutore non è un computer (elettronico): gli astronauti devono "fare gli *automi*" che eseguono un *algoritmo* (quello per la costruzione del filtro - che loro non conoscono perché non sono ingegneri degli impianti di climatizzazione) che da terra viene loro inviato scritto in un *linguaggio* che devono essere in grado di comprendere e le cui azioni devono essere in grado di *attuare*.

Come argomentato da Armoni in [6], i problemi col termine CT nascono quando lo si considera non come un'abbreviazione per indicare "il nucleo scientifico dell'informatica", ma come la definizione di una nuova disciplina. In quest'ottica esemplificativa, l'uso della definizione di Cuny, Snyder e Wing precedentemente riportata può essere utile a patto di sottolineare il ruolo dell'**agente elaboratore di informazioni** (o, per usare un termine più semplice, dell'**esecutore**). Senza l'esecutore e la sua capacità di operare **in modo effettivo**[8], non c'è informatica, ma solo matematica, che in effetti ha risolto problemi da millenni, scoprendo ed applicando in questo lungo percorso l'astrazione, la decomposizione, la ricorsione, e così via… Dello stesso avviso è Alfred Aho, che - in [3], subito prima di enunciare la definizione che ispirerà quella di Cuny, Snyder e Wing - sottolinea come si debba usare il termine "computazione" in congiunzione con l'indicazione di un ben definito modello di calcolo: essendo la computazione un processo definito in termini di un modello sottostante, esso non può essere ignorato.

Inoltre, è proprio l'aver messo l'esecutore al centro del proprio approccio a far sì che l'informatica apporti, rispetto alla matematica, un mutamento radicale di punto di vista. Il cambiamento di

[7] https://www.youtube.com/watch?v=vNaNxwATJqY

[8] Ricordiamo che una procedura è **effettiva** se è costituita da una serie di azioni elementari, eseguibili in modo deterministico ed in tempo finito dall'esecutore.

paradigma concettuale è costituito dal passaggio **dal risolvere i problemi al far risolvere i problemi**. Questa transizione costituisce la «differenza che fa la differenza» [10] e colloca l'informatica come disciplina a sé stante tra le altre scienze.

Questo diverso "sguardo" che l'informatica ha sul mondo è stato inoltre fecondo di conseguenze pratiche.

Ad esempio, la formalizzazione dell'esecutore e delle risorse (tempo, spazio) a sua disposizione per l'effettuazione della computazione ha permesso di caratterizzare in modo esatto la "complessità" della risoluzione dei problemi e di determinarne la gerarchia, dai più semplici ai più difficili. Sulla base di queste conquiste scientifiche, le tecnologie crittografiche permettono di garantire la sicurezza delle transazioni della nostra carta di credito quando eseguiamo acquisti on line.

E ancora, la formalizzazione di come l'esecutore rappresenta i dati a sua disposizione, le relazioni tra essi ed i relativi processi di elaborazione, ha consentito la definizione precisa dei processi con cui l'esecutore può "apprendere" e quindi, ad esempio, fornirci quello specifico annuncio promozionale calibrato sui nostri interessi.

Conoscere i concetti fondamentali della scienza alla base di queste meraviglie della tecnologia digitale ci consente - ancora procedendo per esempi - di discutere con maggiore consapevolezza di vantaggi e criticità del voto elettronico, di capire meglio gli effetti di diffusione di notizie (magari false) sulle reti sociali, e via dicendo. Inoltre, considerando che nell'attuale società digitale la presenza di dispositivi informatici è ubiqua, è necessario per qualunque cittadino che voglia essere a suo agio nella società stessa comprenderne i principi scientifici alla base del loro funzionamento.

In questo l'informatica ha un ruolo non solo descrittivo ed ermeneutico, ma consente di agire sulla realtà. Che si tratti dei corrieri che usano l'informatica per meglio gestire la logistica delle consegne in costante crescita, oppure degli storici che utilizzano tecniche di elaborazione del linguaggio naturale per fare nuove ipotesi su simboli ancora non decifrati, l'informatica è uno strumento necessario per trasformare il mondo secondo i propri scopi, in ogni attività lavorativa e ogni disciplina scientifica.

Un ulteriore elemento caratteristico del particolare approccio che ha l'informatica nel descrivere il mondo è fornito dall'uso dell'astrazione. Si tratta di uno strumento concettuale che tutte le discipline scientifiche usano, ma che nell'informatica assume una natura del tutto particolare, in primo luogo perché **le sue astrazioni sono eseguibili in modo meccanico** [11, 12]. Questo vuol dire che le astrazioni dell'informatica possono essere "animate" senza dover ogni volta costruire una nuova rappresentazione fisica dell'astrazione stessa. In secondo luogo, l'informatica, attraverso i suoi "linguaggi", fornisce un supporto linguistico all'astrazione: opportuni costrutti linguistici permettono di costruire e definire astrazioni in modo sistematico, e di spostarsi in modo uniforme tra i loro vari livelli arrivando senza soluzione di continuità fino all'esecuzione fisica delle astrazioni individuate. In questo l'informatica si distingue dalle altre discipline, che devono ogni volta costruire "ad hoc" un insieme di oggetti materiali che esprimono i fenomeni modellati. E

questa particolarità - che è basata sulla formalizzazione del concetto di esecutore sotto forma della "Macchina Universale di Turing" - fa sì che l'informatica possa costituire un importante ausilio didattico per lo studio delle altre discipline.

3. "Pensare come un informatico"

È importante, a questo punto, andare più in dettaglio e cercare di elencare alcuni degli aspetti peculiari e caratteristici del "pensare come un informatico". Analizzando alcune tra le più importanti definizioni di pensiero computazionale proposte in letteratura (si veda [13] per un'analisi completa), emergono molti elementi in comune.

Tutti coloro che danno una definizione precisa, concordano sul fatto che il pensiero computazionale sia un "**processo mentale**" (o più in generale un "modo di pensare") per "**risolvere problemi**" (problem solving). Ma tutti specificano che non si tratta di una generica risoluzione di problemi: la formulazione del problema e della soluzione devono essere espresse in modo che un "**agente che elabora informazioni**" (essere umano o macchina) sia in grado di comprenderle ed eseguirle.

Gli autori elencano poi quelli che ritengono essere gli elementi costitutivi del pensiero computazionale. Questi elementi sono di tipologie molto diverse (da "abitudini di ragionamento" a "specifici concetti di programmazione" a "competenze trasversali") e sono raggruppati in diverse categorie, ma non c'è consenso sulla loro classificazione.

Proponiamo quattro categorie e, all'interno di ciascuna di esse, elenchiamo alcuni tra gli elementi comuni che riconosciamo nelle definizioni analizzate.

- **Processi mentali**: strategie mentali utili per risolvere problemi
    - *Pensiero algoritmico*: usare il pensiero algoritmico per progettare una sequenza ordinata di passi (istruzioni) per risolvere un problema, ottenere un risultato o portare a termine un compito
    - *Pensiero logico*: usare la logica e il ragionamento per convincersi di qualcosa, stabilire e controllare fatti
    - *Scomposizione di problemi*: dividere un problema complesso in semplici sottoproblemi, risolubili in modo più semplice; modularizzare; usare il ragionamento composizionale
    - *Astrazione*: liberarsi dei dettagli inutili per concentrarsi sulle informazioni / idee rilevanti
    - *Riconoscimento di pattern*: individuare regolarità/schemi ricorrenti nei dati e nei problemi
    - *Generalizzazione*: usare le regolarità riconosciute per fare previsioni o per risolvere problemi più generali
- **Metodi**: approcci operativi utilizzati dagli informatici
    - *Automazione*: automatizzare soluzioni; usare un computer o una macchina per eseguire compiti ripetitivi o noiosi

    - *Raccolta, analisi e rappresentazione dei dati*: raccogliere informazioni e dati, interpretarli trovando schemi ricorrenti, rappresentarli in maniera appropriata; memorizzare, recuperare e aggiornare dati
    - *Parallelizzazione*: eseguire compiti simultaneamente per raggiungere un obiettivo comune, pensare "in parallelo"
    - *Simulazione*: rappresentare dati e processi (del mondo reale) tramite modelli; eseguire esperimenti su tali modelli
    - *Valutazione*: analizzare le soluzioni implementate per giudicarne la bontà, in particolare per ciò che riguarda la loro effettività e la loro efficienza in termini di tempo impiegato o di spazio occupato
    - *Programmazione*: usare alcuni concetti di base della programmazione (cicli, eventi, istruzioni condizionali, operatori logici…)
- **Pratiche**: usate tipicamente nell'implementazione di soluzioni informatiche
    - *Sperimentare, iterare, fare "tinkering"*: nelle metodologie di sviluppo software incrementali e iterative, un progetto viene sviluppato attraverso ripetizioni di un ciclo "progetta-costruisci-verifica", costruendo in modo incrementale il risultato finale; fare "tinkering" significa costruire qualcosa usando un processo per prove ed errori, imparare dal gioco, dall'esplorazione e dalla sperimentazione
    - *Testare e correggere gli errori (debug)*: verificare che le soluzioni funzionino provandole concretamente; trovare e risolvere i problemi (i "bug") in una soluzione o in un programma
    - *Riuso e remix*: costruire la propria soluzione basandosi su / utilizzando anche codice, progetti o idee già esistenti
- **Competenze trasversali**: modi di vedere e operare nel mondo; utili competenze per la vita favorite dal "pensare come un informatico"
    - *Creare*: progettare e costruire artefatti, usare la computazione per essere creativi ed esprimere se stessi
    - *Comunicare e collaborare*: connettersi con gli altri e lavorare insieme con un obiettivo comune per creare qualcosa e per ottenere una soluzione migliore
    - *Riflettere, imparare, fare meta-cognizione*: usare l'informatica per riflettere e comprendere gli aspetti computazionali del mondo
    - *Tollerare l'ambiguità*: avere a che fare con problemi reali, aperti e non totalmente specificati a priori
    - *Perseverare quando si ha a che fare con problemi difficili*: essere a proprio agio nel lavorare con problemi difficili/complessi, essere determinati, resilienti e tenaci

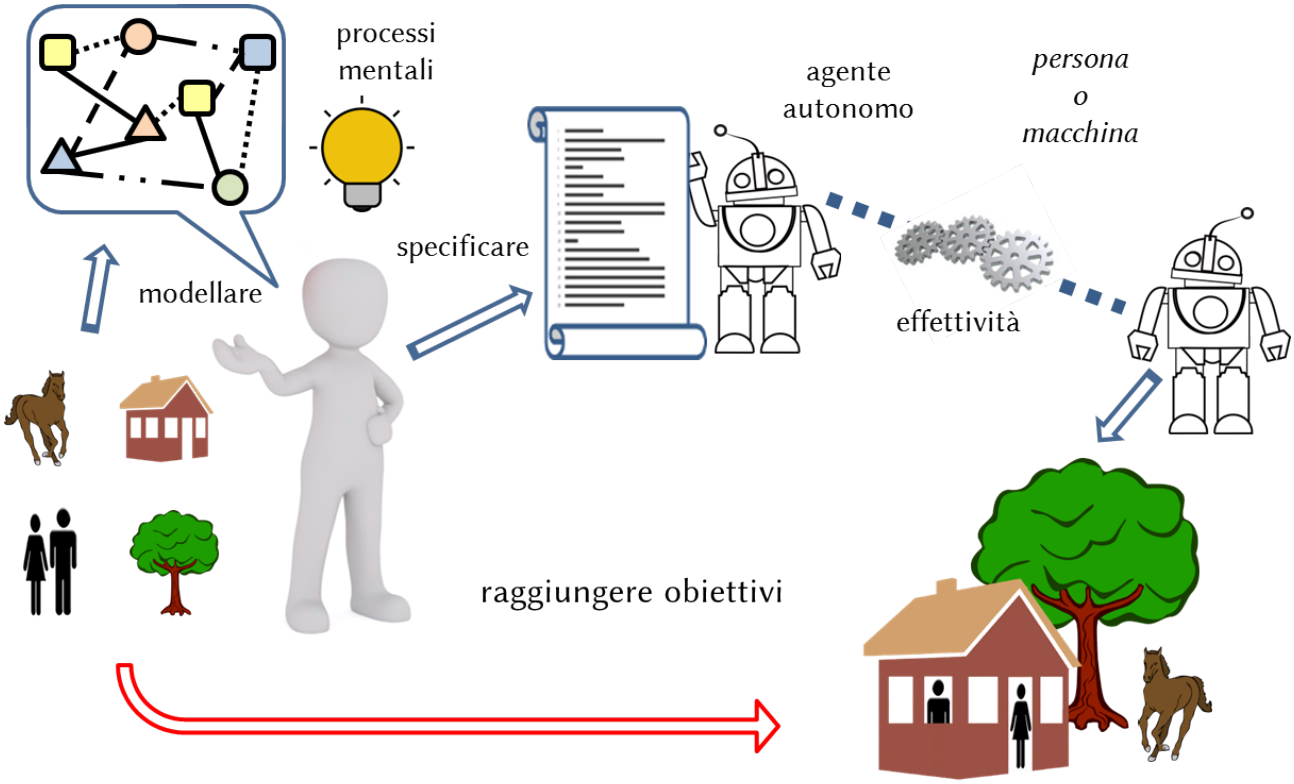
Sempre nell'ottica di una spiegazione della natura dell'informatica, va osservato che la formulazione originale della Wing del termine "pensiero computazionale" è un po' restrittiva. Lei stessa aveva infatti chiarito in [2]: «*La mia interpretazione dei termini "problema" e "soluzione" è vasta. Non mi riferisco solo a problemi ben definiti da un punto di vista matematico e le cui soluzioni sono formulabili in modo completo, p.es., mediante una prova, un algoritmo, o un*

programma, ma anche a problemi del mondo reale le cui soluzioni possono essere fornite sotto forma di un sistema software grande e complesso».

Ma anche in questa spiegazione viene usato il termine "problema", che intrinsecamente evoca il significato di "qualcosa che deve essere risolto".

Per superare questa difficoltà, rimanendo in linea con l'obiettivo esplicativo, si può fornire una definizione più generale: "**Il pensiero computazionale è l'insieme dei processi mentali usati per modellare una *situazione* e specificare i modi mediante i quali un agente elaboratore di informazioni può operare in modo effettivo all'interno della situazione stessa per raggiungere uno o più obiettivi forniti *dall'esterno***" [14].

Questa formulazione contiene due differenze importanti. La prima è far riferimento ad una situazione (un contesto) in cui l'esecutore opera, invece che ad un problema che l'agente deve risolvere. La seconda è chiarire che l'esecutore non si assegna autonomamente gli obiettivi da raggiungere, ma li ottiene dall'esterno. In tal modo la definizione è anche più vicina alle recenti caratterizzazioni del concetto di computazione come processo illimitato (p.es., [15]).



4. Informatica a Scuola

Ricapitolando, le ultime considerazioni hanno messo in luce due aspetti fondamentali:

- l'informatica ha una sua dignità come disciplina scientifica di base, con il suo insieme di concetti e metodi stabili ed indipendenti;
- l'informatica ha un importante valore come disciplina trasversale, in grado di fornire un approccio che contribuisce ad una migliore comprensione delle altre discipline [16].

Questi due aspetti hanno un ruolo fondamentale anche nella discussione sull'introduzione dell'Informatica a scuola.

Per quel che riguarda il primo punto, già nel primo articolo di Jeannette Wing [1], e poi in vari lavori successivi, il pensiero computazionale è stato indicato come "quarta abilità di base" da insegnare oltre alle classiche "leggere, scrivere e far di conto"[9]. Tale convinzione emerge anche in lavori meno recenti: per esempio, nelle parole di Forsythe citate in apertura.

Le "Indicazioni nazionali per il curricolo" [17] sottolineano che il compito del primo ciclo d'istruzione è quello di promuovere "l'alfabetizzazione culturale e sociale di base" per acquisire linguaggi e codici della nostra cultura (allargata alle altre culture conviventi e all'uso consapevole dei nuovi media). Viene specificato che tale alfabetizzazione *include* l'alfabetizzazione strumentale ("leggere, scrivere e far di conto"), e la potenzia tramite i linguaggi e i saperi delle discipline. Questo dunque inquadra le "tre abilità di base" - il cui apprendimento era nei secoli scorsi il traguardo dell'alfabetizzazione - come *strumenti* per un più alto compito: imparare a comprendere il contesto sociale in cui lo studente si trova a vivere. Strumenti che, nel complesso mondo di oggi, hanno bisogno dei linguaggi e dei saperi specifici di diverse discipline per poter fornire una formazione di base adeguata. L'informatica è, a nostro avviso, una di tali discipline.

Riguardo al secondo punto, notiamo che la definizione da noi proposta si configura come una generalizzazione di quella di Cuny, Snyder e Wing, e racchiude scenari di estremo interesse per la scuola e l'istruzione: la simulazione dei fenomeni. In quest'ambito si costruiscono e manipolano rappresentazioni visibili di leggi fisiche e/o di fenomeni sociali/naturali più che risolvere problemi in sé e per sé. In altre parole, si modella una situazione e si esplorano le sue possibili evoluzioni allo scopo di acquisirne una migliore comprensione.

Infine, il pensiero computazionale viene spesso legato a competenze trasversali (es. problem solving "generale", ragionamento logico [13]) o alle "competenze del 21° secolo"[10] o alle "competenze chiave e di cittadinanza" quali collaborazione, comunicazione, pensiero critico, creatività, perseveranza e resilienza, eccetera: alcune di queste competenze sono infatti citate nelle definizioni di pensiero computazionale.
Sebbene riteniamo che "pensare come un informatico" possa favorire lo sviluppo di queste competenze (pensiero computazionale → competenze trasversali), osserviamo che tali competenze possono essere sviluppate anche mediante altri approcci. Notiamo però la tendenza di alcuni insegnanti (comprensibile, visto che sono familiari con esse) a utilizzare l'implicazione nel senso opposto (competenze trasversali → pensiero computazionale), sostenendo di insegnare il pensiero computazionale quando stanno insegnando, per esempio, a perseverare o a essere

[9] In inglese se ne parla come "the fourth R", dal momento che le tre abilità di base sono "Reading, wRiting, aRithmetic".
[10] http://www3.weforum.org/docs/WEFUSA_NewVisionforEducation_Report2015.pdf

creativi. Si tratta chiaramente di una fallacia di ragionamento, che porta a snaturare l'indissolubile legame tra il pensiero computazionale e gli aspetti distintivi e peculiari della disciplina che lo genera: l'Informatica.

Un aspetto rilevante nel dibattito di questi anni riguarda proprio il rapporto tra pensiero computazionale e una di queste competenze trasversali: la creatività.

Sir Ken Robinson, nella prefazione a [18], ci ricorda che la tecnologia ha da sempre aiutato l'uomo ad espandere il proprio corpo e la propria mente. La specie umana si distingue proprio per il suo potere di astrazione e immaginazione. Per far lavorare questa immaginazione, ed essere quindi creativi, sono necessari materiali e strumenti, e il risultato dipende molto anche da quali di questi sono disponibili. Quando si parla di creatività - sottolinea Mitch Resnick [18] - spesso si pensa all'espressione artistica, oppure alla "Creatività con la C maiuscola" dei grandi inventori o personaggi le cui opere sono esposte nei musei.

Ma la creatività non è solo degli artisti, ma anche degli scienziati che fanno scoperte, dei medici che diagnosticano malattie o degli imprenditori che sviluppano nuovi prodotti o strategie di mercato. E, più in generale, entra in gioco quando si sviluppano idee che sono utili a noi stessi o agli altri nella vita di tutti i giorni: probabilmente non si tratta di scoperte innovative nel panorama mondiale, ma lo sono per noi. In altre parole, si tratta della "creatività con la c minuscola", di cui tutti possono essere capaci, a patto che venga coltivata negli studenti fin da quando sono piccoli.

Poiché l'informatica rende possibile costruire rappresentazioni e soluzioni per le più diverse situazioni, anche quelle fisicamente irrealizzabili, limitati solo dalla nostra immaginazione, essa costituisce una potentissima palestra per esercitare la creatività, e dunque ha un alto valore educativo. In un mondo in cui le informazioni sono facilmente reperibili, e i computer sono sempre più bravi a risolvere compiti noiosi e ripetitivi, diventa ancor più importante educare gli studenti a trovare soluzioni innovative, piuttosto che a esercitarsi nell'applicazione di procedure mnemoniche o standardizzate.

Concludiamo questa sezione con una riflessione sulle possibili concezioni errate che rischiano di generarsi in seguito all'introduzione dell'Informatica a scuola. Si tratta di opinioni "opposte" l'una con l'altra, ma di cui vediamo segnali nelle discussioni odierne.

La prima riguarda le idee sbagliate e stereotipate sull'informatica: per molti programmare e ragionare da informatici rimane appannaggio di poche persone che hanno quello che gli americani chiamano il "gene del geek[11]" [19] o appartengono a specifiche categorie: individui con un unico interesse, asociali, competitivi, maschi [20]. Forse proprio a causa di queste idee alcuni (p.es. sui media) tendono a sottolineare che "l'obiettivo non è quello di insegnare informatica", come se questo fosse un obiettivo negativo o non meritevole di essere perseguito. Analoghe concezioni stereotipate sono presenti già da tempo per ciò che riguarda la matematica, e sono tra le cause di insuccesso degli studenti [21].

[11] Termine usato nell'inglese colloquiale per indicare le persone particolarmente abili con le tecnologie digitali.

La seconda, all'opposto, riguarda l'idea che associa l'Informatica solo alle attività introduttive giocose e ludiche - svolte magari esclusivamente durante gli eventi quali "La settimana europea del codice" o "L'ora del codice". Tali eventi e attività, fondamentali per portare l'attenzione sull'importanza dell'apprendimento dell'Informatica, e per interessare e coinvolgere gli studenti, devono essere seguite però da un percorso sistematico e strutturato (adeguato all'età dei discenti) di introduzione ai concetti fondamentali della disciplina.

5. Conclusioni

Nell'ultimo decennio l'espressione "pensiero computazionale" è stata utilizzata per indicare il "nucleo scientifico dell'informatica", con lo scopo di sensibilizzare le persone sull'importanza che, nella moderna società digitale e dell'informazione, l'apprendimento di tali concetti riveste a tutti i livelli di istruzione.

Il grande risalto che ha avuto questo tema ha portato però a delle concezioni errate su cosa il "pensiero computazionale" realmente sia, prima fra tutte l'idea che tale espressione indichi una disciplina nuova e distinta dall'Informatica.

Invece di cercare di definire in modo preciso che cosa sia il pensiero computazionale - azione che può aver contribuito ad aumentare la confusione sul tema - siamo risaliti alle motivazioni storiche e culturali che hanno reso l'informatica una scienza nuova e indipendente dalla matematica e abbiamo messo in evidenza quali aspetti peculiari la caratterizzano e distinguono tutt'oggi. In particolare, abbiamo argomentato sull'importanza di riferirsi ad un esecutore. L'informatico non risolve problemi, ma li fa risolvere a una "macchina": scrive un algoritmo, e lo traduce in un linguaggio ad essa comprensibile, di modo che essa possa interpretare ed eseguire le istruzioni che le vengono fornite.

Dalla nostra discussione, risulta chiaro che l'Informatica ha una sua dignità sia come disciplina scientifica di base, da insegnare così come si fa con le altre scienze, sia come disciplina trasversale. Sia perché il suo potere di "automatizzare le astrazioni" permette di "dar vita" a modelli di fenomeni altrimenti impossibili da studiare in azione, sia perché le pratiche messe in atto dagli informatici possono essere un valido mezzo per sviluppare competenze trasversali fondamentali nella società di oggi. Prima fra tutte, la creatività: indispensabile per utilizzare le tecnologie in maniera attiva e per emergere come esseri umani protagonisti in un mondo sempre più automatizzato.

Riferimenti bibliografici

[1] Wing, J. (2006). "Computational thinking", *Communications of the ACM*, 49(3), 33-35.

[2] Wing, J. (2011). "Research Notebook: Computational Thinking – What and Why?", *The LINK. The Magazine of Carnegie Mellon University's School of Computer Science*.

https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why (ultimo accesso ottobre 2017)

[3] Aho, A. (2011). “Ubiquity symposium: Computation and Computational Thinking”, *Ubiquity*, 2011(Jan).

[4] Knuth, D.E. (1974). “Computer Science and Its Relation to Mathematics”, *The American Mathematical Monthly*, 81(4), 323-343.

[5] Forsythe, G.E. (1968). “What To Do Till The Computer Scientist Comes”, *The American Mathematical Monthly*, 75(5), 454-462.

[6] Armoni, M. (2015). “Computer Science, Computational Thinking, Programming, Coding: The Anomalies of Transitivity in K-12 Computer Science Education”, *ACM Inroads*, 7(4), 24-27.

[7] Denning, P.J. (2017). “Computational Thinking in Science”, *American Scientist*, 105, 3-17.

[8] Polanyi, M. (1966). *The Tacit Dimension*, The University of Chicago Press.

[9] Denning, P.J., Tedre, M., Yongpradit, P. (2017). “Misconceptions about computer science”, *Communications of the ACM*, 60(3), 31-33.

[10] Bateson, G. (1972). *Form, Substance and Difference*, in *Steps to an Ecology of Mind*, University of Chicago Press.

[11] Wing, J. (2008). “Computational thinking and thinking about computing”, *Philosophical Transactions of The Royal Society A*, 366, 3717-3725.

[12] Enrico Nardelli (2017). “La grande bellezza dell’informatica”, http://link-and-think.blogspot.it/2017/05/la-grande-bellezza-dellinformatica.html (ultimo accesso ottobre 2017).

[13] Corradini, I., Lodi, M., Nardelli, E. (2017). “Conceptions and Misconceptions about Computational Thinking among Italian Primary School Teachers”, *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*, 136-144.

[14] Nardelli, E. (2017). “Do we really need computational thinking?”, *Comm. ACM*, in corso di pubblicazione.

[15] van Leeuwen, J., Wiedermann, J. (2012). “Computation as an unbounded process”, *Theoretical Computer Science*, 429, 202-212.

[16] Denning, P.J., Rosenbloom, P.S. (2009). “Computing: The Fourth Great Domain of Science”, *Communications of the ACM*, 52(9), 27-29.

[17] MIUR. (2012). *Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione*. http://www.indicazioninazionali.it (ultimo accesso ottobre 2017)

[18] Resnick, M. (2017). *Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play*, MIT Press.

[19] Ahadi, A., Lister, R. (2013). “Geek genes, prior knowledge, stumbling points and learning edge momentum: parts of the one elephant?”, *Proceedings of the ninth annual international ACM conference on International computing education research (ICER '13)*, 123-128.

[20] Lewis, C.M., Anderson, R.E., Yasuhara, K. (2016). “I Don't Code All Day: Fitting in Computer Science When the Stereotypes Don't Fit”, *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*, 23-32.

[21] Caponi, B., Falco, G., Focchiatti, R., Cornoldi, C., Lucangeli, D. (2006). *Didattica metacognitiva della matematica*, Erickson.

Michael Lodi è dottorando in Informatica presso l’Università di Bologna. Le sue ricerche riguardano la didattica dell’informatica e il pensiero computazionale, con particolare interesse per la formazione dei docenti e per le competenze trasversali che l’apprendimento dell’informatica può sviluppare. Tiene laboratori di Informatica Creativa per insegnanti presenti e futuri, e per bambini all’interno del CoderDojo Bologna. È laureato in Informatica e abilitato all’insegnamento di Informatica nella scuola secondaria di secondo grado. Email: michael.lodi@unibo.it , Twitter: @ldomhl

Simone Martini è ordinario di Informatica e Direttore del Dip. di Informatica-Scienza e Ingegneria dell’Alma Mater Studiorum-Università di Bologna. Laureato in Scienze dell’Informazione e Dottore di Ricerca in Informatica, ha insegnato anche a Pisa e Udine, e trascorso periodi di ricerca alla Stanford University, l’École normale supérieure di Parigi, l’Université Paris 13, l’University of California at Santa Cruz. Le sue ricerche riguardano i fondamenti logici dei linguaggi di programmazione, e la storia e la filosofia dell’informatica. Email: simone.martini@unibo.it

Enrico Nardelli è ordinario di Informatica nel Dip. di Matematica dell’Univ. Roma Tor Vergata. Laureato in Ingegneria Elettronica, la sua ricerca attuale verte sull'integrazione tra aspetti tecnologici e sociali/personali dei Sistemi Informativi e sull'istruzione informatica nella scuola. Coordina il progetto MIUR-CINI “Programma il Futuro” per la diffusione nelle scuole della formazione di base sull’informatica. È presidente di Informatics Europe, membro dello ACM Europe Council e del Consiglio Direttivo del CINI. È stato presidente del GRIN, l'associazione dei docenti universitari di Informatica. e-mail: nardelli@mat.uniroma2.it , Twitter: @enriconardelli